Raising the Cost of Malicious AI-Powered Image Editing

Hadi Salman [*1] **Alaa Khaddaj** [*1] **Guillaume Leclerc** [*1] **Andrew Ilyas** [1] **Aleksander Mądry** [1]

Abstract

We present an approach to mitigating the risks of malicious image editing posed by large diffusion models. The key idea is to *immunize* images so as to make them resistant to manipulation by these models. This immunization relies on injection of imperceptible adversarial perturbations designed to disrupt the operation of the targeted diffusion models, forcing them to generate unrealistic images. We provide two methods for crafting such perturbations, and then demonstrate their efficacy. Finally, we discuss a policy component necessary to make our approach fully effective and practical—one that involves the organizations developing diffusion models, rather than individual users, to implement (and support) the immunization process.[1]

1. Introduction

Large diffusion models such as DALL·E 2 (Ramesh et al., 2022) and Stable Diffusion (Rombach et al., 2022) are known for their ability to produce high-quality photorealistic images, and can be used for a variety of image synthesis and editing tasks. However, the ease of use of these models has raised concerns about their potential abuse, e.g., by creating inappropriate or harmful digital content. For example, a malevolent actor might download photos of people posted online and edit them maliciously using an off-the-shelf diffusion model (as in Figure 1 top).

How can we address these concerns? First, it is important to recognize that it is, in some sense, impossible to completely eliminate such malicious image editing. Indeed, even without diffusion models in the picture, malevolent actors can still use tools such as Photoshop to manipulate existing images, or even synthesize fake ones entirely from scratch. The key new problem that large generative models introduce is that these actors can now create realistic edited images with *ease*, i.e., without the need for specialized skills or expensive equipment. This realization motivates us to ask:

How can we raise the cost of malicious (AI-powered) image manipulation?

In this paper, we put forth an approach that aims to alter the economics of AI-powered image editing. At the core of our approach is the idea of image *immunization*—that is, making a specific image resistant to AI-powered manipulation by adding a carefully crafted (imperceptible) perturbation to it. This perturbation would disrupt the operation of a diffusion model, forcing the edits it performs to be unrealistic (see Figure 1). In this paradigm, people can thus continue to share their (immunized) images as usual, while getting a layer of protection against undesirable manipulation.

We demonstrate how one can craft such imperceptible perturbations for large-scale diffusion models and show that they can indeed prevent realistic image editing. We then discuss in Section 5 complementary technical and policy components needed to make our approach fully effective and practical.

2. Preliminaries

We start by providing an overview of diffusion models as well as of the key concept we will leverage: adversarial attacks.

2.1. Diffusion Models

Diffusion models have emerged recently as powerful tools for generating realistic images (Sohl-Dickstein et al., 2015; Ho et al., 2020). These models excel especially at generating and editing images using textual prompts, and currently surpass other image generative models such as GANs (Goodfellow et al., 2014) in terms of the quality of produced images.

Diffusion process. At their core, diffusion models employ a stochastic differential process called the *diffusion process* (Sohl-Dickstein et al., 2015). This process allows

[*]Equal contribution [1]MIT. Correspondence to: Hadi Salman <hady@mit.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]Code is available at https://github.com/MadryLab/photoguard.

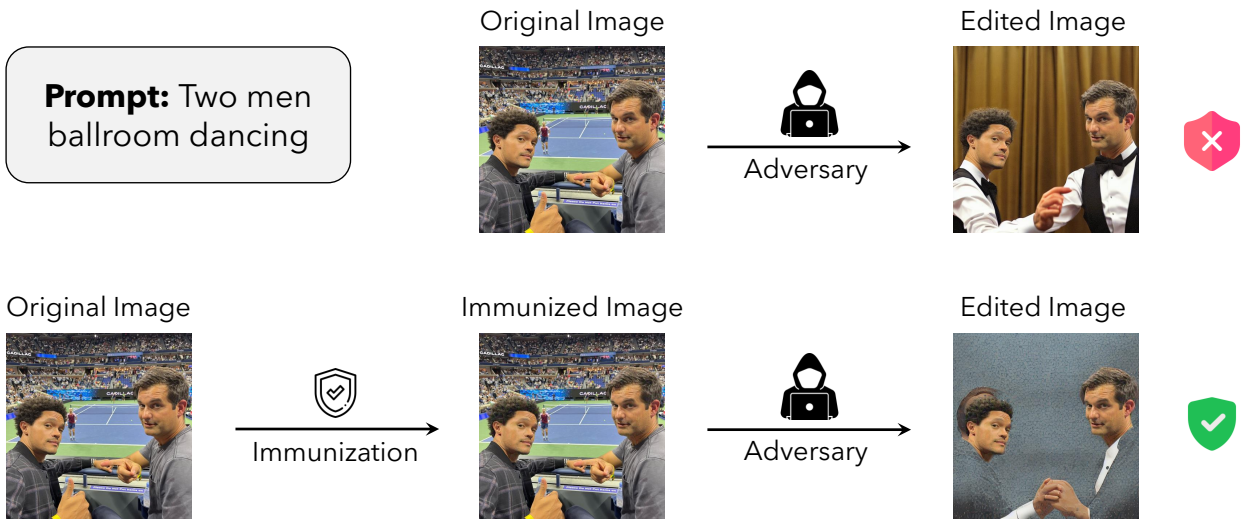

Figure 1: *Overview of our framework.* An adversary seeks to modify an image found online. The adversary describes via a textual prompt the desired changes and then uses a diffusion model to generate a realistic image that matches the prompt (top). By immunizing the original image before the adversary can access it, we disrupt their ability to successfully perform such edits (bottom).

us to view the task of (approximate) sampling from a distribution of real images $q(\cdot)$ as a series of *denoising* problems. More precisely, given a sample $\mathbf{x}_0 \sim q(\cdot)$, the diffusion process incrementally adds noise to generate samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$ for T steps, where $\mathbf{x}_{t+1} = a_t \mathbf{x}_t + b_t \boldsymbol{\varepsilon}_t$, and $\boldsymbol{\varepsilon}_t$ is sampled from a Gaussian distribution[2]. Note that, as a result, the sample $\mathbf{x}_T$ starts to follow a standard normal distribution $\mathcal{N}(0, \mathbf{I})$ when $T \to \infty$. Now, if we reverse this process and are able to sample $\mathbf{x}_t$ given $\mathbf{x}_{t+1}$, i.e., *denoise* $\mathbf{x}_{t+1}$, we can ultimately generate new samples from $q(\cdot)$. This is done by simply starting from $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ (which corresponds to T being sufficiently large), and iteratively denoising these samples for T steps, to produce a new image $\tilde{\mathbf{x}} \sim q(\cdot)$.

The element we need to implement this process is thus to learn a neural network $\boldsymbol{\varepsilon}_\theta$ that "predicts" given $\mathbf{x}_{t+1}$ the noise $\boldsymbol{\varepsilon}_t$ added to $\mathbf{x}_t$ at each time step t. Consequently, this *denoising model* $\boldsymbol{\varepsilon}_\theta$ is trained to minimize the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, \mathbf{x}_0, \boldsymbol{\varepsilon} \sim \mathcal{N}(0,1)} \left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_{t+1}, t)\|_2^2 \right], \quad (1)$$

where t is sampled uniformly over the T time steps. We defer discussion of details to Appendix B and refer the reader to (Weng, 2021) for a more in-depth treatment of diffusion models.

[2]Here, a_t and b_t are the parameters of the distribution $q(\mathbf{x}_{t+1}|\mathbf{x}_t)$. Details are provided in Appendix B.

Latent diffusion models (LDMs). Our focus will be on a specific class of diffusion models called the *latent diffusion models* (LDMs) (Rombach et al., 2022)[3]. These models apply the diffusion process described above in the *latent space* instead of the input (image) space. As it turned out, this change enables more efficient training and faster inference, while maintaining high quality generated samples.

Training an LDM is similar to training a standard diffusion model and differs mainly in one aspect. Specifically, to train an LDM, the input image $\mathbf{x}_0$ is first mapped to its latent representation $\mathbf{z}_0 = \mathcal{E}(\mathbf{x}_0)$, where $\mathcal{E}$ is a given encoder. The diffusion process then continues as before (just in the *latent* space) by incrementally adding noise to generate samples $\mathbf{z}_1, \ldots, \mathbf{z}_T$ for T steps, where $\mathbf{z}_{t+1} = a_t \mathbf{z}_t + b_t \boldsymbol{\varepsilon}_t$, and $\boldsymbol{\varepsilon}_t$ is sampled from a Gaussian distribution. Finally, the denoising network $\boldsymbol{\varepsilon}_\theta$ is then learned analogously to as before but, again, now in the latent space, by minimizing the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{t, \mathbf{z}_0, \boldsymbol{\varepsilon} \sim \mathcal{N}(0,1)} \left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\mathbf{z}_{t+1}, t)\|_2^2 \right] \quad (2)$$

Once the denoising network $\boldsymbol{\varepsilon}_\theta$ is trained, the same generative process can be applied as before, starting from a random vector in the latent space, to obtain a latent representation $\tilde{\mathbf{z}}$ of the (new) generated image. This representation is then decoded into an image $\tilde{\mathbf{x}} = \mathcal{D}(\tilde{\mathbf{z}}) \sim q(\cdot)$, using

[3]Our methodology can be adjusted to other diffusion models. Our focus on LDMs is motivated by the fact that all popular open-sourced diffusion models are of this type.

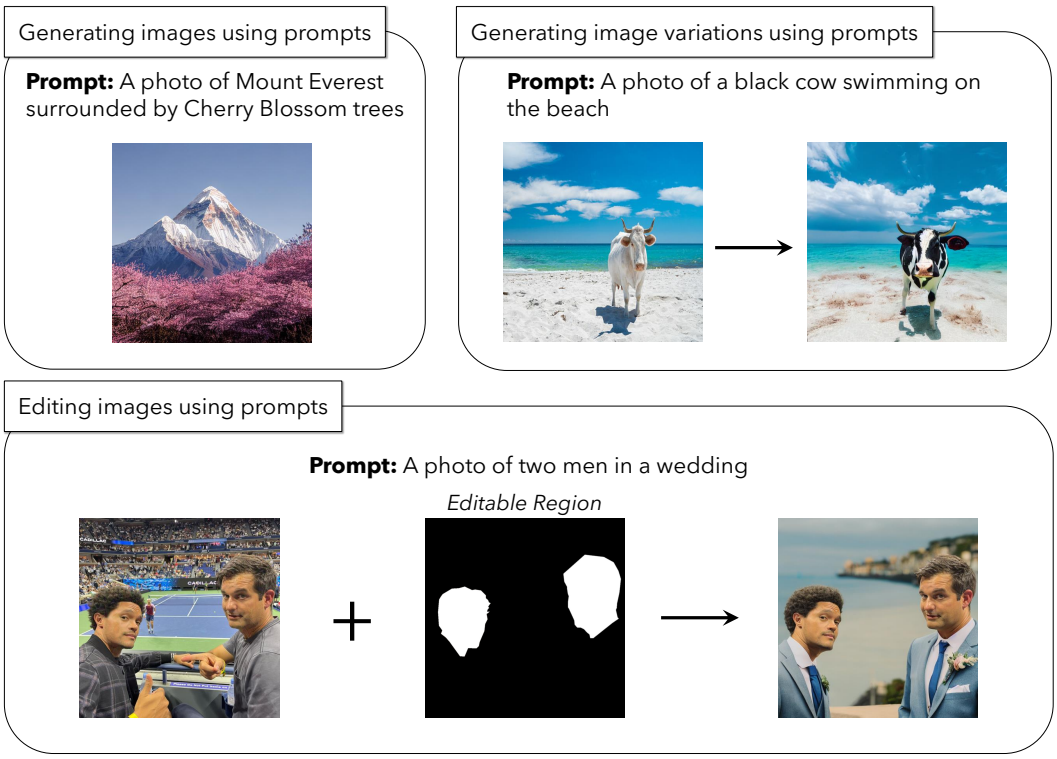

Figure 2: Diffusion models offer various capabilities, such as (1) generating images using text prompts (top left), (2) generating variations of an input image using text prompts (top right), and (3) editing images using text prompts (bottom).

the corresponding decoder $\mathcal{D}$.

Prompt-guided sampling using an LDM. An LDM by default generates a random sample from the distribution of images $q(\cdot)$ it was trained on. However, it turns out one can also guide the sampling using natural language. This can be accomplished by combining the latent representation $\mathbf{z}_T$ produced during the diffusion process with the embedding of the user-defined textual prompt t.[4] The denoising network ε_θ is applied to the combined representation for T steps, yielding $\tilde{\mathbf{z}}$ which is then mapped to a new image using the decoder $\mathcal{D}$ as before.

LDMs capabilities. LDMs turn out to be powerful text-guided image generation and editing tools. In particular, LDMs can be used not only for generating images using textual prompts, as described above, but also for generating textual prompt–guided variations of an image or edits of a specific part of an image (see Figure 2). The latter two capabilities (i.e., generation of image variations and image editing) requires a slight modification of the generative process described above. Specifically, to modify or edit a given image $\mathbf{x}$, we condition the generative process on this image. That is, instead of applying, as before, our generative process of T denoising steps to a random vector in the latent space, we apply it to the latent representation obtained from running the latent diffusion process on our image $\mathbf{x}$. To edit only part of the image we additionally condition the process on freezing the parts of the image that were to remain unedited.

[4]Conditioning on the text embedding happens at every stage of the generation process. See (Rombach et al., 2022) for more details.

2.2. Adversarial Attacks

For a given computer vision model and an image, an *adversarial example* is an imperceptible perturbation of that image that manipulates the model's behavior (Szegedy et al., 2014; Biggio et al., 2013). In image classification, for example, an adversary can construct an adversarial example for a given image $\mathbf{x}$ that makes it classified as a specific target label y_{targ} (different from the true label). This construction is achieved by minimizing the loss of a classifier f_θ with respect to that image:

$$\delta_{adv} = \arg\min_{\delta \in \Delta} \mathcal{L}(f_\theta(\mathbf{x}+\delta), y_{targ}). \tag{3}$$

Here, Δ is a set of perturbations that are small enough that they are imperceptible—a common choice is to constrain the adversarial example to be close (in ℓ_p distance) to the

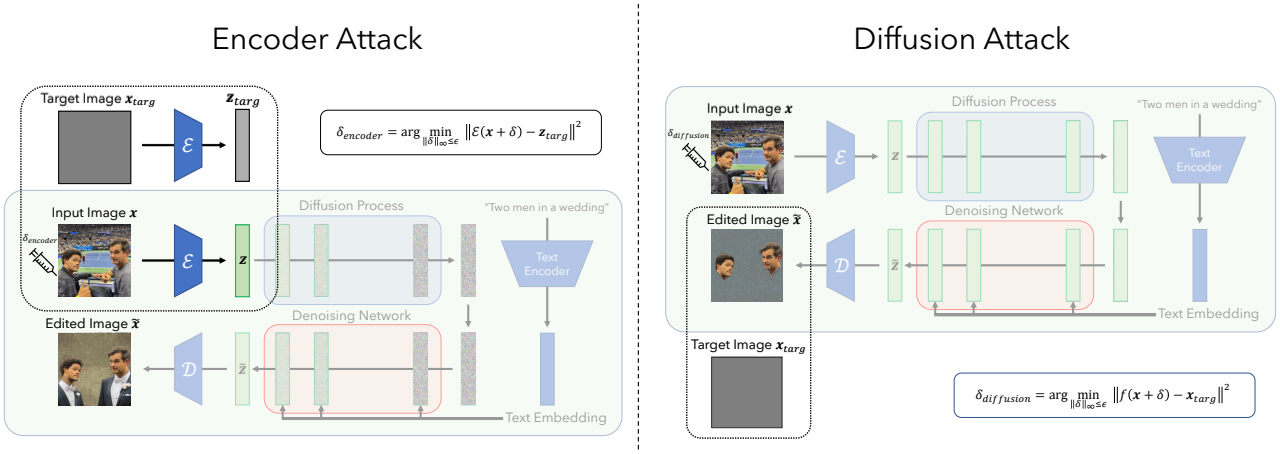

Figure 3: *Overview of our proposed attacks*. When applying the *encoder* attack (left), our goal is to map the representation of the original image to the representation of a target image (gray image). Our (more complex) *diffusion* attack (right), on the other hand, aims to break the diffusion process by manipulating the whole process to generate image that resembles a given target image (gray image).

original image, i.e., $\Delta = \{\delta : \|\delta\|_p \le \epsilon\}$. The canonical approach to constructing an adversarial example is to solve the optimization problem (3) via projected gradient descent (PGD) (Nesterov, 2003; Madry et al., 2018).

3. Adversarially Attacking Latent Diffusion Models

We now describe our approach to immunizing images, i.e., making them harder to manipulate using latent diffusion models (LDMs). At the core of our approach is to leverage techniques from the adversarial attacks literature (Szegedy et al., 2014; Madry et al., 2018; Akhtar et al., 2021) and add adversarial perturbations (see Section 2.1) to immunize images. Specifically, we present two different methods to execute this strategy (see Figure 3): an *encoder attack*, and a *diffusion attack*.

Encoder attack. Recall that an LDM, when applied to an image, first encodes the image using an encoder $\mathcal{E}$ into a latent vector representation, which is then used to generate a new image (see Section 2). The key idea behind our encoder attack is now to disrupt this process by forcing the encoder to map the input image to some "bad" representation. To achieve this, we solve the following optimization problem using projected gradient descent (PGD):

$$\delta_{encoder} = \arg\min_{\|\delta\|_\infty \le \epsilon} \|\mathcal{E}(\mathbf{x}+\delta) - \mathbf{z}_{targ}\|_2^2, \quad (4)$$

where $\mathbf{x}$ is the image to be immunized, and $\mathbf{z}_{targ}$ is some target latent representation (e.g., $\mathbf{z}_{targ}$ can be the representation, produced using encoder $\mathcal{E}$, of a gray image). Solutions to this optimization problem yield small, imperceptible perturbations $\delta_{encoder}$ which, when added to the original image, result in an (immunized) image that is similar to the (gray) target image from the LDM's encoder perspective. This, in turn, causes the LDM to generate an irrelevant or unrealistic image. An overview of this attack is shown in Figure 3 (left)[5].

Diffusion attack. Although the encoder attack is effective at forcing the LDM to generate images that are unrelated to the immunized ones, we still expect the LDM to use the textual prompt. For example, as shown in the encoder attack diagram in Figure 3, editing an immunized image of two men using the prompt "Two men in a wedding" still results in a generated image with two men wearing wedding suits, even if the image will contain some visual artifacts indicating that it has been manipulated. Can we disturb the diffusion process even further so that the diffusion model "ignores" the textual prompt entirely and generates a more obviously manipulated image?

It turns out that we are able to do so by using a more complex attack, one where we target the diffusion process itself instead of just the encoder. In this attack, we perturb the input image so that the *final* image generated by the LDM is a specific target image (e.g., random noise or gray image). Specifically, we generate an adversarial perturbation $\delta_{diffusion}$ by solving the following optimization problem (again via PGD):

$$\delta_{diffusion} = \arg\min_{\|\delta\|_\infty \le \epsilon} \|f(\mathbf{x}+\delta) - \mathbf{x}_{targ}\|_2^2. \quad (5)$$

Above, f is the LDM, $\mathbf{x}$ is the image to be immunized, and $\mathbf{x}_{targ}$ is the target image to be generated. An overview

[5] See Algorithm 1 for the details of the encoder attack.

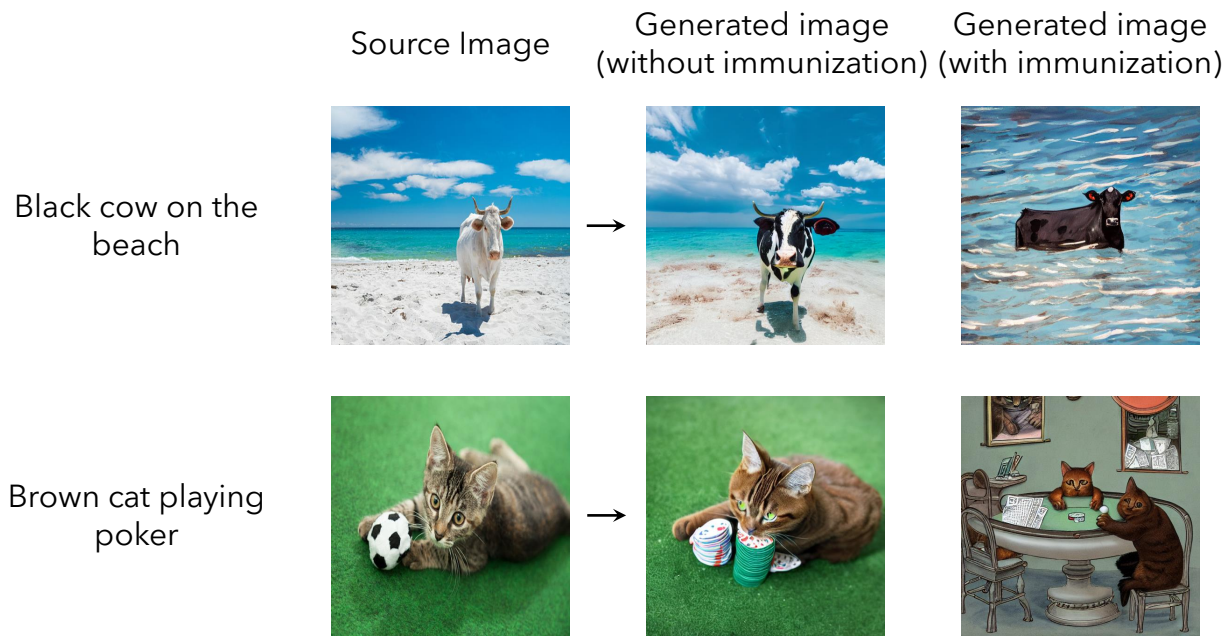

Figure 4: Given a source image (e.g., image of a white cow on the beach) and a textual prompt (e.g., "black cow on the beach"), the SDM can generate a realistic image matching the prompt while still similar to the original image (middle column). However, when the source image is immunized, the SDM fails to do so (right-most column). More examples are in Appendix C.

of this attack is depicted in Figure 3 (right)[6]. As we already mentioned, this attack targets the full diffusion process (which includes the text prompt conditioning), and tries to nullify not only the effect of the immunized image, but also that of the text prompt itself. Indeed, in our example (see Figure 3 (right)) no wedding suits appear in the edited image whatsoever.

It is worth noting that this approach, although more powerful than the encoder attack, is harder to execute. Indeed, to solve the above problem (5) using PGD, one needs to backpropagate through the full diffusion process (which, as we recall from Section 2.1, includes repeated application of the denoising step). This causes memory issues even on the largest GPU we used[7]. To address this challenge, we backpropagate through only a few steps of the diffusion process, instead of the full process, while achieving adversarial perturbations that are still effective. We defer details of our attacks to Appendix A.

4. Results

In this section, we examine the effectiveness of our proposed immunization method.

[6]See Algorithm 2 for the details of the diffusion attack.
[7]We used an A100 with 40 GB memory.

Setup. We focus on the Stable Diffusion Model (SDM) v1.5 (Rombach et al., 2022), though our methods can be applied to other diffusion models too. In each of the following experiments, we aim to disrupt the performance of SDM by adding imperceptible noise (using either of our proposed attacks)—i.e., applying our immunization procedure—to a variety of images. The goal is to force the model to generate images that are unrealistic and unrelated to the original (immunized) image. We evaluate the performance of our method both qualitatively (by visually inspecting the generated images) and quantitatively (by examining the image quality using standard metrics). We defer further experimental details to Appendix A.

4.1. Qualitative Results

Immunizing against generating image variations. We first assess whether we can disrupt the SDM's ability to generate realistic variations of an image based on a given textual prompt. For example, given an image of a white cow on the beach and a prompt of "black cow on the beach", the SDM should generate a realistic image of a *black* cow on the beach that looks similar to the original one (cf. Figure 4). Indeed, the SDM is able to generate such images. However, when we immunize the original images (using the encoder attack), the SDM fails to generate a realistic variation—see Figure 4.

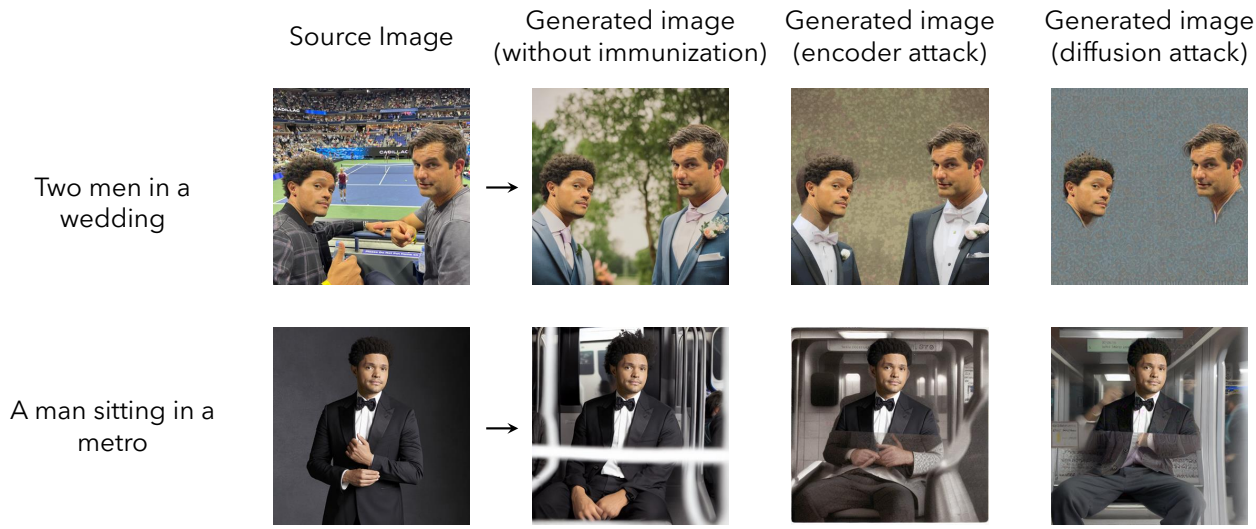

Figure 5: Given a source image (e.g., image of two men watching a tennis game) and a textual prompt (e.g., "two men in a wedding"), the SDM can edit the source image to match the prompt (second column). However, when the source image is immunized using the *encoder attack*, the SDM fails to do so (third column). Immunizing using the *diffusion attack* further reduces the quality of the edited image (forth column). More examples are in Appendix C.

Immunizing against image editing. Now we consider the more challenging task of disrupting the ability of SDMs to edit images using textual prompts. The process of editing an image using an SDM involves inputting the image, a mask indicating which parts of the image should be edited, and a text prompt guiding how the rest of the image should be manipulated. The SDM then generates an edited version based on that prompt. An example can be seen in Figure 2, where an image of two men watching a tennis game is transformed to resemble a wedding photo. This corresponded to inputting the original image, a binary mask excluding from editing only the men's heads, and the prompt "A photo of two men in a wedding." However, when the image is immunized (using either encoder or diffusion attacks), the SDM is unable to produce realistic image edits (cf. Figure 5). Furthermore, the diffusion attack results in more unrealistic images than the encoder attack.

4.2. Quantitative Results

Image quality metrics. Figures 4 and 5 indicate that, as desired, edits of immunized images are noticeably different from those of non-immunized images. To quantify this difference, we generate 60 different edits of a variety of images using different prompts, and then compute several metrics capturing the similarity between resulting edits of immunized versus non-immunized images[8]: FID (Heusel et al., 2017), PR (Sajjadi et al., 2018), SSIM (Wang et al., 2004), PSNR, VIFp (Sheikh & Bovik, 2006), and FSIM (Zhang et al., 2011)[9]. The better our immunization method is, the less similar the edits of immunized images are to those of non-immunized images.

The similarity scores, shown in Table 1, indicate that applying either of our immunization methods (encoder or diffusion attacks) indeed yields edits that are different from those of non-immunized images (since, for example, FID is far from zero for both of these methods). As a baseline, we consider a naive immunization method that adds uniform random noise (of the same intensity as the perturbations used in our proposed immunization method). This method, as we verified, is not effective at disrupting the SDM, and yields edits almost identical to those of non immunized images. Indeed, in Table 1, the similarity scores of this baseline indicate closer edits to non-immunized images compared to both of our attacks.

Image-prompt similarity. To further evaluate the quality of the generated/edited images after immunization (using diffusion attack), we measure the similarity between the edited images and the textual prompt used to guide this edit, with and without immunization. The fact that the SDM uses the textual prompt to guide the generation of an image indicates that the similarity between the generated image and the prompt should be high in the case of no immunization. However, after immunization (using the diffusion attack), the similarity should be low, since the immuniza-

[8]We use the implementations provided in: https://github.com/photosynthesis-team/piq.

[9]We report additional metrics in Appendix C.1.

Method	FID ↓	PR ↑	SSIM ↑	PSNR ↑	VIFp ↑	FSIM ↑
Immunization baseline (Random noise)	82.57	1.00	0.75 ± 0.13	19.21 ± 4.00	0.43 ± 0.13	0.83 ± 0.08
Immunization (Encoder attack)	130.6	0.95	0.58 ± 0.11	14.91 ± 2.78	0.30 ± 0.10	0.73 ± 0.08
Immunization (Diffusion attack)	**167.6**	**0.87**	$\mathbf{0.50 \pm 0.09}$	$\mathbf{13.58 \pm 2.23}$	$\mathbf{0.24 \pm 0.09}$	$\mathbf{0.69 \pm 0.06}$

Table 1: We report various image quality metrics measuring the similarity between edits originating from immunized vs. non-immunized images. We observe that edits of immunized images are substantially different from those generated from the original (notn-immunized) images. Note that the arrows next to the metrics denote increasing image similarity. Since our goal is to make the edits as different as possible from the original edits in the presence of no immunization, then lower image similarity is better. Confidence intervals denote one standard deviation over 60 images. Additional metrics are in Appendix C.1.

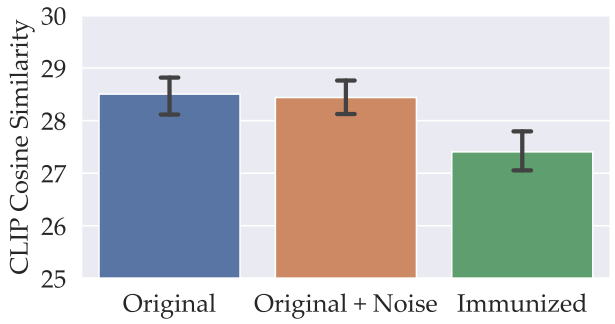

Figure 6: *Image-prompt similarity*. We plot the cosine similarity between the CLIP embeddings of the generated images and the text prompts, with and without immunization, as well as with a baseline immunization of adding small random noise to the original image. Error bars denote the interquartile range (IQR) over 60 runs.

tion process disrupts the full diffusion process, and forces the diffusion model to ignore the prompt during generation. We use the same 60 edits as in our previous experiment, and we extract—using a pretrained CLIP model (Radford et al., 2021)—the visual embeddings of these images and the textual prompts used to generate them. We then compute the cosine similarity between these two embeddings. As show in Figure 6, the immunization process decreases the similarity between the generated images and the textual prompts to generated them, as expected.

5. A Techno-Policy Approach to Mitigation of AI-Powered Editing

In the previous sections we have developed an immunization procedure that, when applied to an image, protects the immunized version of that image from realistic manipulation by a given diffusion model. Our immunization procedure has, however, certain important limitations. We now discuss these limitations as well as a combination of technical and policy remedies needed to obtain a fully effective approach to raising the cost of malicious AI-powered image manipulation.

(Lack of) robustness to transformations. One of the limitations of our immunization method is that the adversarial perturbation that it relies on may be ineffective after the immunized image is subjected to image transformations and noise purification techniques. For instance, malicious actors could attempt to remove the disruptive effect of that perturbation by cropping the image, adding filters to it, applying a rotation, or other means. This problem can be addressed, however, by leveraging a long line of research on creating *robust* adversarial perturbations, i.e., adversarial perturbations that can withstand a broad range of image modifications and noise manipulations (Eykholt et al., 2018; Kurakin et al., 2016; Athalye et al., 2018; Brown et al., 2018).

Forward-compatibility of the immunization. While the immunizing adversarial perturbations we produce might be effective at disrupting the current generation of diffusion-based generative models, they are not guaranteed to be effective against the future versions of these models. Indeed, one could hope to rely here on the so-called *transferability* of adversarial perturbations (Papernot et al., 2016; Liu et al., 2017), but *no* perturbation will be perfectly transferable.

To truly address this limitation, we thus need to go beyond purely technical methods and encourage—or compel—via policy means a collaboration between organizations that develop large diffusion models, end-users, as well as data hosting and dissemination platforms. Specifically, this collaboration would involve the developers providing APIs that allow the users and platforms to immunize their images against manipulation by the diffusion models the developers create. Importantly, these APIs should guarantee "forward compatibility", i.e., effectiveness of the offered immunization against models developed in the future. This can be accomplished by planting, when training such future models, the current immunizing adversarial perturbations

as backdoors. (Observe that our immunization approach can provide *post-hoc* "backward compatibility" too. That is, one can create immunizing adversarial perturbations that are effective for models that were *already released*.)

It is important to point out that we are leveraging here an incentive alignment that is fundamentally different to the one present in more typical applications of adversarial perturbations and backdoor attacks. In particular, the "attackers" here—that is, the parties that create the adversarial perturbations/execute the backdoor attack—are the same parties that develop the models being attacked. This crucial difference is, in particular, exactly what helps remedy the forward compatibility challenges that turns out to be crippling, e.g., in the context of "unlearnable" images creation (i.e., creation of images that are immune to being leveraged by, e.g., facial recognition models) (Radiya-Dixit et al., 2021).

6. Related Work

Data misuse after model training. Recent advances in ML-powered image generation and editing have raised concerns about the potential misuse of personal data for generating fake images. This issue arose first in the context of the development of generative adversarial networks (GANs) for image generation and editing (Goodfellow et al., 2014; Mirza & Osindero, 2014; Salimans et al., 2016; Isola et al., 2017; Zhu et al., 2017; Zhang et al., 2017; Karras et al., 2018; Brock et al., 2019; Karras et al., 2019), and led to research on methods for defending against such manipulation, such as attacking the GAN itself (Ruiz et al., 2020a;b; Sun et al., 2021). This problem became exacerbated recently with the advent of (publicly available) diffusion models (Rombach et al., 2022; Ramesh et al., 2022). Indeed, one can now easily describe in text how one wants to manipulate an image, and immediately get the result of an impressive quality (see Figure 2) that significantly outperforms previous methods, such as GANs.

Deepfake detection. A line of work related to ours aims to *detect* fake images rather than *prevent* their generation. Deepfake detection methods include analyzing the consistency of facial expressions and identifying patterns or artifacts in the image that may indicate manipulation, and training machine learning models to recognize fake images (Korshunov & Marcel, 2018; Afchar et al., 2018; Nguyen et al., 2019; Mirsky & Lee, 2021; Rossler et al., 2019; Durall et al., 2019; Li et al., 2020a;b; Bonettini et al., 2021). While some deepfake detection methods are more effective than others, no single method is foolproof. A potential way to mitigate this shortcoming could involve development of so-called watermarking methods (Cox et al., 1997; Neekhara et al., 2022). These methods aim to ensure that it is easy to detect that a given output has been produced using a generative model—such watermarking approaches have been recently developed for a related context of large language models (Kirchenbauer et al., 2023). Still, neither deepfake detection nor watermarking methods could protect images from being manipulated in the first place. A manipulated image can hence cause harm before being flagged as fake. Also, given that our work is complementary to deepfake detection and watermarking methods, it could, in principle, be combined with them.

Data misuse during model training. The abundance of readily available data on the Internet has played a significant role in recent breakthroughs in deep learning, but has also raised concerns about the potential misuse of such data when training models. Therefore, there has been an increasing interest in protection against unauthorized data exploitation, e.g., by designing unlearnable examples (Huang et al., 2021; Fu et al., 2021). These methods propose adding imperceptible backdoor signals to user data before uploading it online, so as to prevent models from fruitfully utilizing this data. However, as pointed out by Radiya-Dixit et al. (2021), these methods can be circumvented, often simply by waiting until subsequently developed models can avoid being fooled by the planted backdoor signal.

7. Conclusion

In this paper, we presented a method for raising the difficulty of using diffusion models for malicious image manipulation. Our method involves "immunizing" images through addition imperceptible adversarial perturbations. These added perturbations disrupt the inner workings of the targeted diffusion model and thus prevent it from producing realistic modifications of the immunized images.

We also discussed the complementary policy component that will be needed to make our approach fully effective. This component involves ensuring cooperation of the organizations developing diffusion-based generative models in provisioning APIs that allow users to immunize their images to manipulation by such models (and the future versions of thereof).

8. Acknowledgements

Work supported in part by the NSF grants CNS-1815221 and DMS-2134108, and Open Philanthropy. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0015. Work partially done on the MIT Supercloud compute cluster (Reuther et al., 2018).

References

Afchar, D., Nozick, V., Yamagishi, J., and Echizen, I. Mesonet: a compact facial video forgery detection network. In *2018 IEEE international workshop on information forensics and security (WIFS)*, pp. 1–7. IEEE, 2018.

Akhtar, N., Mian, A., Kardan, N., and Shah, M. Threat of adversarial attacks on deep learning in computer vision: Survey. 2021.

Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

Balanov, A., Schwartz, A., Moshe, Y., and Peleg, N. Image quality assessment based on dct subband similarity. In *IEEE International Conference on Image Processing (ICIP)*, 2015.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases (ECML-KDD)*, 2013.

Bonettini, N., Cannas, E. D., Mandelli, S., Bondi, L., Bestagini, P., and Tubaro, S. Video face manipulation detection through ensemble of cnns. In *2020 25th international conference on pattern recognition (ICPR)*, pp. 5012–5019. IEEE, 2021.

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch, 2018.

Cox, I. J., Kilian, J., Leighton, F. T., and Shamoon, T. Secure spread spectrum watermarking for multimedia. *IEEE transactions on image processing*, 6(12):1673–1687, 1997.

Durall, R., Keuper, M., Pfreundt, F.-J., and Keuper, J. Unmasking deepfakes with simple features. *arXiv preprint arXiv:1911.00686*, 2019.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Tramer, F., Prakash, A., Kohno, T., and Song, D. Physical adversarial examples for object detectors. *CoRR*, 2018.

Fu, S., He, F., Liu, Y., Shen, L., and Tao, D. Robust unlearnable examples: Protecting data privacy against adversarial learning. In *International Conference on Learning Representations*, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *neural information processing systems (NeurIPS)*, 2014.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems (NeurIPS)*, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. Unlearnable examples: Making personal data unexploitable. In *ICLR*, 2021.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *conference on computer vision and pattern recognition (CVPR)*, 2017.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. 2023.

Korshunov, P. and Marcel, S. Deepfakes: a new threat to face recognition? assessment and detection. *arXiv preprint arXiv:1812.08685*, 2018.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Li, L., Bao, J., Zhang, T., Yang, H., Chen, D., Wen, F., and Guo, B. Face x-ray for more general face forgery detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5001–5010, 2020a.

Li, Y., Yang, X., Sun, P., Qi, H., and Lyu, S. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3207–3216, 2020b.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. In

International Conference on Learning Representations (ICLR), 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Mirsky, Y. and Lee, W. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54 (1):1–41, 2021.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Neekhara, P., Hussain, S., Zhang, X., Huang, K., McAuley, J., and Koushanfar, F. Facesigns: semi-fragile neural watermarks for media authentication and countering deepfakes. *arXiv preprint arXiv:2204.01960*, 2022.

Nesterov, Y. *Introductory Lectures on Convex Optimization*. 2003.

Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T., and Nahavandi, S. Deep learning for deepfakes creation and detection. 2019.

Papernot, N., McDaniel, P., and Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. In *ArXiv preprint arXiv:1605.07277*, 2016.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *arXiv preprint arXiv:2103.00020*, 2021.

Radiya-Dixit, E., Hong, S., Carlini, N., and Tramèr, F. Data poisoning won't save you from facial recognition. arXiv, 2021.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Reisenhofer, R., Bosse, S., Kutyniok, G., and Wiegand, T. A haar wavelet-based perceptual similarity index for image quality assessment. 2018.

Reuther, A., Kepner, J., Byun, C., Samsi, S., Arcand, W., Bestor, D., Bergeron, B., Gadepally, V., Houle, M., Hubbell, M., Jones, M., Klein, A., Milechin, L., Mullen, J., Prout, A., Rosa, A., Yee, C., and Michaleas, P. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6. IEEE, 2018.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., and Nießner, M. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1–11, 2019.

Ruiz, N., Adel Bargal, S., and Sclaroff, S. Disrupting deepfakes: Adversarial attacks against conditional image translation networks and facial manipulation systems. 2020a.

Ruiz, N., Bargal, S., and Sclaroff, S. Protecting against image translation deepfakes by leaking universal perturbations from black-box neural networks. 2020b.

Sajjadi, M. S. M., Bachem, O., Lučić, M., Bousquet, O., and Gelly, S. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *neural information processing systems (NeurIPS)*, 2016.

Sheikh, H. and Bovik, A. Image information and visual quality. 2006.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Sun, H., Zhu, T., Zhang, Z., Xiong, D. J., Zhou, W., et al. Adversarial attacks against deep generative models on data: a survey. *arXiv preprint arXiv:2112.00247*, 2021.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. Image quality assessment: from error visibility to structural similarity. 2004.

Weng, L. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

Xue, W., Zhang, L., Mou, X., and Bovik, A. C. Gradient magnitude similarity deviation: A highly efficient perceptual image quality index. 2014.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017.

Zhang, L. and Li, H. Sr-sim: A fast and high performance iqa index based on spectral residual. In *2012 19th IEEE International Conference on Image Processing*, 2012.

Zhang, L., Zhang, L., Mou, X., and Zhang, D. Fsim: A feature similarity index for image quality assessment. 2011.

Zhang, L., Shen, Y., and Li, H. Vsi: A visual saliency-induced index for perceptual image quality assessment. 2014.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *international conference on computer vision(ICCV)*, 2017.

A. Experimental Setup

A.1. Details of the diffusion model we used

In this paper, we used the open-source stable diffusion model hosted on the Hugging Face[10]. We use the hyperparameters presented in Table 2 to generated images from this model. For a given image on which we want to test our immunization method, we first search for a good random seed that leads to a realistic modification of the image given some textual prompt. Then we use the same seed when editing the immunized version of the same image using the diffusion model. This ensures that the immunized image is modified in the same way as the original image, and that the resulting non-realistic edits are due to immunization and not to random seed.

Table 2: Hyperparameters used for the Stable Diffusion model.

height	width	guidance_scale	num_inference_steps	eta
512	512	7.5	100	1

A.2. Our attacks details

Throughout the paper, we use two different attacks: an encoder attack and a diffusion attack. These attacks are described in the main paper, and are summarized here in Algorithm 1 and Algorithm 2, respectively. For both of the attacks, we use the same set of hyperparameters shown in Table 3. The choice of ϵ was such that it is the large enough to disturb the image, but small enough to not be noticeable by the human eye.

Table 3: Hyperparameters used for the adversarial attacks.

Norm	ϵ	step size	number of steps
ℓ_∞	16/255	2/255	200

Algorithm 1 Encoder Attack on a Stable Diffusion Model

1: **Input:** Input image $\mathbf{x}$, target image $\mathbf{x}_{targ}$, Stable Diffusion model encoder $\mathcal{E}$, perturbation budget ϵ, step size k, number of steps N.
2: Compute the embedding of the target image: $\mathbf{z}_{targ} \leftarrow \mathcal{E}(\mathbf{x}_{targ})$
3: Initialize adversarial perturbation $\delta_{encoder} \leftarrow 0$, and immunized image $\mathbf{x}_{im} \leftarrow \mathbf{x}$
4: **for** $n = 1 \dots N$ **do**
5: Compute the embedding of the immunized image: $\mathbf{z} \leftarrow \mathcal{E}(\mathbf{x}_{im})$
6: Compute mean squared error: $l \leftarrow \|\mathbf{z}_{targ} - \mathbf{z}\|_2^2$
7: Update adversarial perturbation: $\delta_{encoder} \leftarrow \delta_{encoder} + k \cdot \text{sign}(\nabla_{\mathbf{x}_{im}} l)$
8: $\delta_{encoder} \leftarrow \text{clip}(\delta_{encoder}, -\epsilon, \epsilon)$
9: Update the immunized image: $\mathbf{x}_{im} \leftarrow \mathbf{x}_{im} - \delta_{encoder}$
10: **end for**
11: **Return:** $\mathbf{x}_{im}$

B. Extended Background for Diffusion Models

Overview of the diffusion process. At their heart, diffusion models leverage a statistical concept: the diffusion process (Sohl-Dickstein et al., 2015; Ho et al., 2020). Given a sample $\mathbf{x}_0$ from a distribution of real images $q(\cdot)$, the diffusion process works in two steps: a forward step and a backward step. During the forward step, Gaussian noise is added to the sample $\mathbf{x}_0$ over T time steps, to generate increasingly noisier versions $\mathbf{x}_1, \dots, \mathbf{x}_T$ of the original sample $\mathbf{x}_0$, until the sample is equivalent to an isotropic Gaussian distribution. During the backward step, the goal is to reconstruct the original sample $\mathbf{x}_0$ by iteratively denoising the noised samples $\mathbf{x}_T, \dots, \mathbf{x}_1$. The power of the diffusion models stems from the

[10]This model is available on: https://huggingface.co/runwayml/stable-diffusion-v1-5.

Algorithm 2 Diffusion Attack on a Stable Diffusion Model

Input: Input image $\mathbf{x}$, target image $\mathbf{x}_{targ}$, Stable Diffusion model f, perturbation budget ϵ, step size k, number of steps N.
Initialize adversarial perturbation $\delta_{diffusion} \leftarrow 0$, and immunized image $\mathbf{x}_{im} \leftarrow \mathbf{x}$
for $n = 1 \dots N$ **do**
  Generate an image using diffusion model: $\mathbf{x}_{out} \leftarrow f(\mathbf{x}_{im})$
  Compute mean squared error: $l \leftarrow \|\mathbf{x}_{targ} - \mathbf{x}_{out}\|_2^2$
  Update adversarial perturbation: $\delta_{diffusion} \leftarrow \delta_{diffusion} + k \cdot \text{sign}(\nabla_{\mathbf{x}_{im}} l)$
  $\delta_{diffusion} \leftarrow \text{clip}(\delta_{diffusion}, -\epsilon, \epsilon)$
  Update the immunized image: $\mathbf{x}_{im} \leftarrow \mathbf{x}_{im} - \delta_{diffusion}$
end for
Return: $\mathbf{x}_{im}$

ability to learn the backward process using neural networks. This allows to generate new samples from the distribution $q(\cdot)$ by first generating a random Gaussian sample, and then passing it through the "neural" backward step.

Forward process. During the forward step, Gaussian noise is iteratively added to the original sample $\mathbf{x}_0$. The forward process $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is assumed to follow a Markov chain, i.e. the sample at time step t depends only on the sample at the previous time step. Furthermore, the variance added at a time step t is controlled by a schedule of variances $\{\beta_t\}_{t=1}^T$[11].

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}); \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \tag{6}$$

Backward process. At the end of the forward step, the sample $\mathbf{x}_T$ looks as if it is sampled from an isotropic Gaussian $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. Starting from this sample, the goal is to recover $\mathbf{x}_0$ by iteratively removing the noise using neural networks. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is referred to as the reverse process, and is also assumed to be a Markov chain.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t); \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{7}$$

Training a diffusion model. At its heart, diffusion models are trained in a way similar to Variational Autoencoders, i.e. by optimizing a variational lower bound. Additional tricks are employed to make the process faster. For an extensive derivation, refer to (Weng, 2021).

$$\mathbb{E}_{q(\mathbf{x}_0)}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] = L_{VLB} \tag{8}$$

Latent Diffusion Models (LDMs). In this paper, we focus on a specific class of diffusion models, namely LDMs, which was proposed in (Rombach et al., 2022) as a model that applies the diffusion process described above in a *latent space* instead of the image space. This enables efficient training and inference of diffusion models.

To train an LDM, the input image $\mathbf{x}_0$ is first mapped to a latent representation $\mathbf{z}_0 = \mathcal{E}(\mathbf{x}_0)$, where $\mathcal{E}$ is an image encoder. This input representation $\mathbf{z}_0$ is then passed to the diffusion process to obtain a denoised $\tilde{\mathbf{z}}$. The generated image $\tilde{\mathbf{x}}$ is then obtained by decoding $\tilde{\mathbf{z}}_0$ using a decoder $\mathcal{D}$, i.e. $\tilde{\mathbf{x}} = \mathcal{D}(\tilde{\mathbf{z}})$.

C. Additional Results

C.1. Additional quantitative results

We presented in Section 4 several metrics to assess the similarity between the images generated with and without immunization. Here, we report in Table 4 additional metrics to evaluate this: SR-SIM (Zhang & Li, 2012), GMSD (Xue et al., 2014), VSI (Zhang et al., 2014), DSS (Balanov et al., 2015), and HaarPSI (Reisenhofer et al., 2018). Similarly, we indicate

[11]The values of a_t and b_t from the main paper correspond to $a_t = \sqrt{1-\beta_t}$ and $b_t = \beta_t$

for each metric whether a higher value corresponds to higher similarity (using ↑), or contrariwise (using ↓). We again observe that applying the encoder attack already decreases the similarity between the generated images with and without immunization, and applying the diffusion attack further decreases the similarity.

Table 4: Additional similarity metrics for Table 1. Errors denote standard deviation over 60 images.

Method	**SR-SIM** ↑	**GMSD** ↓	**VSI** ↑	**DSS** ↑	**HaarPSI** ↑
Immunization baseline (Random noise)	0.91 ± 0.04	0.20 ± 0.06	0.94 ± 0.03	0.35 ± 0.18	0.52 ± 0.15
Immunization (Encoder attack)	0.86 ± 0.05	0.26 ± 0.05	0.90 ± 0.03	0.19 ± 0.09	0.35 ± 0.11
Immunization (Diffusion attack)	$\mathbf{0.84 \pm 0.05}$	$\mathbf{0.27 \pm 0.04}$	$\mathbf{0.89 \pm 0.03}$	$\mathbf{0.17 \pm 0.08}$	$\mathbf{0.31 \pm 0.08}$

C.2. Generating Image Variations using Textual Prompts

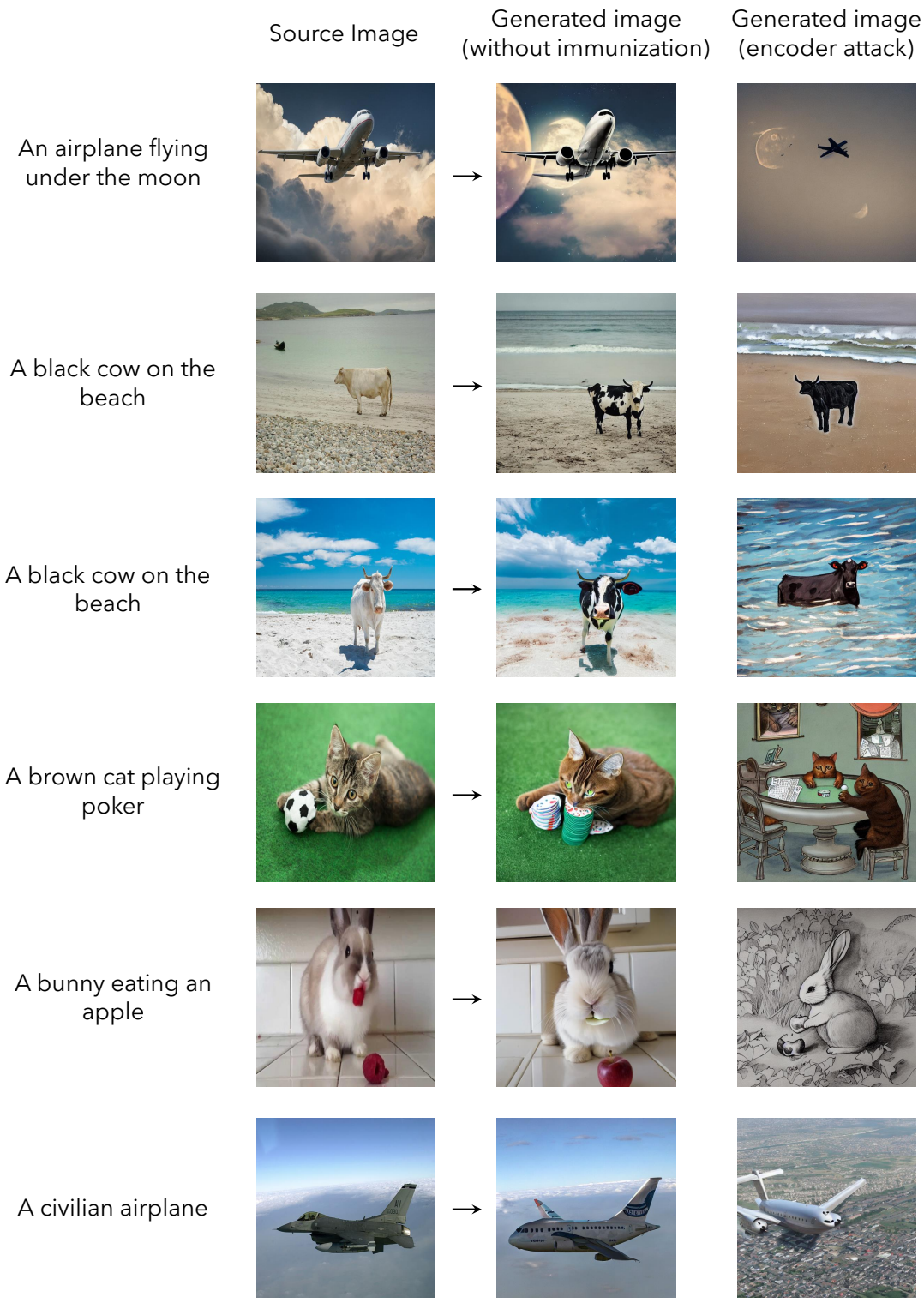

Figure 7: Immunization against generating prompt-guided image variations.

C.3. Image Editing via Inpainting

Figure 8: Immunization against image editing via prompt-guided inpainting.

Figure 9: Immunization against image editing via prompt-guided inpainting.

Figure 10: Immunization against image editing via prompt-guided inpainting.

Figure 11: Immunization against image editing via prompt-guided inpainting.

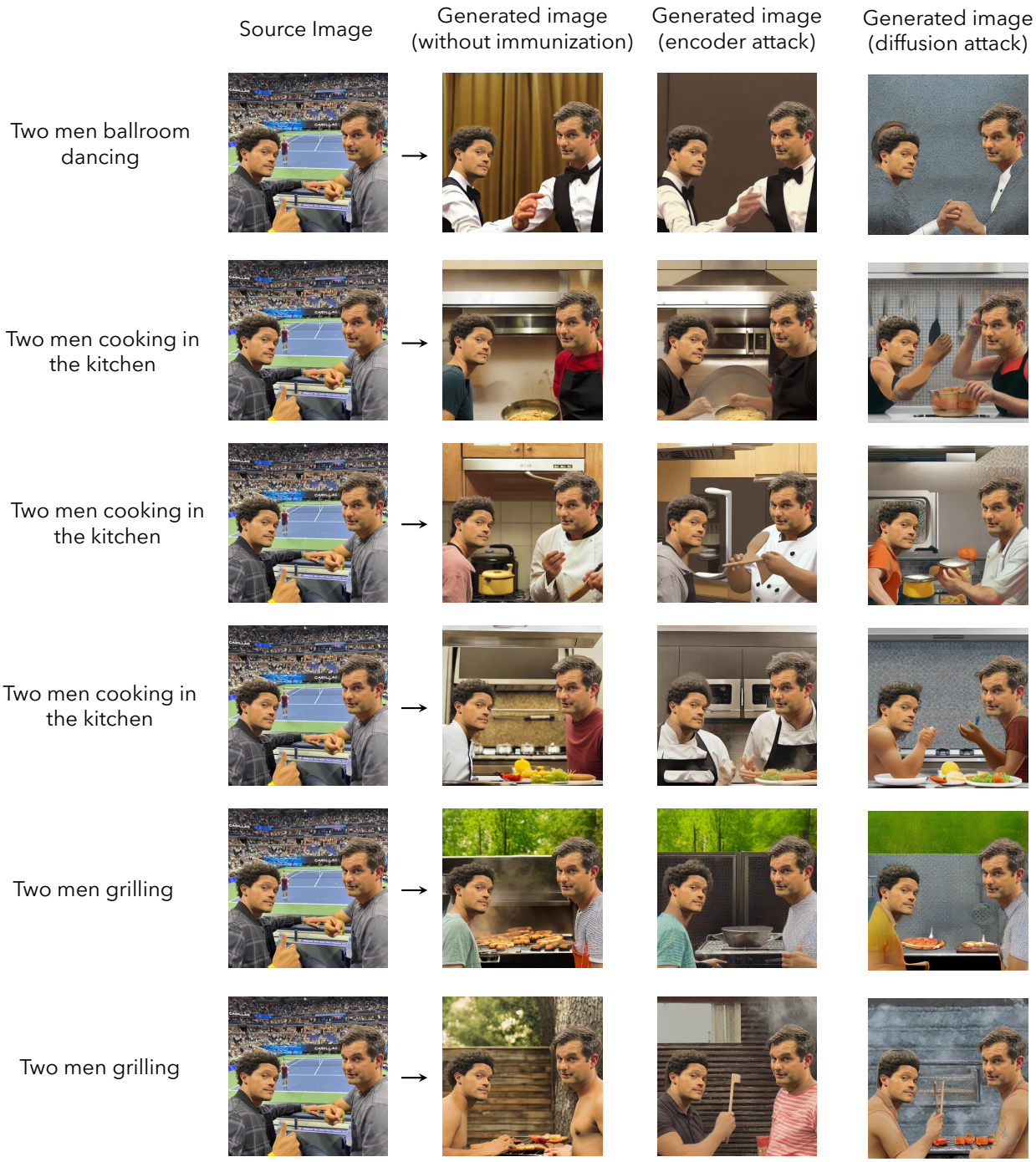

Figure 12: Immunization against image editing via prompt-guided inpainting.

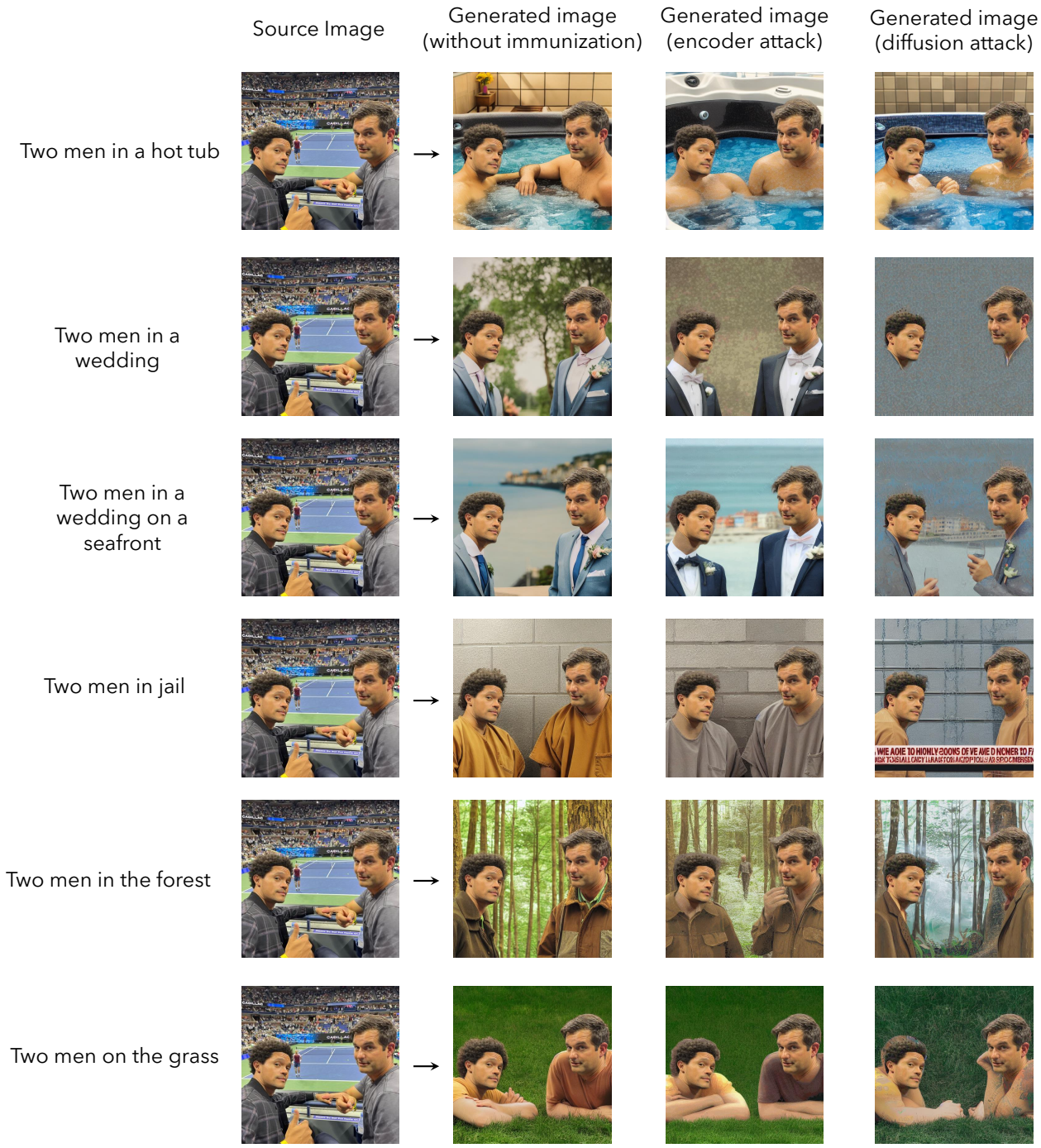

Figure 13: Immunization against image editing via prompt-guided inpainting.

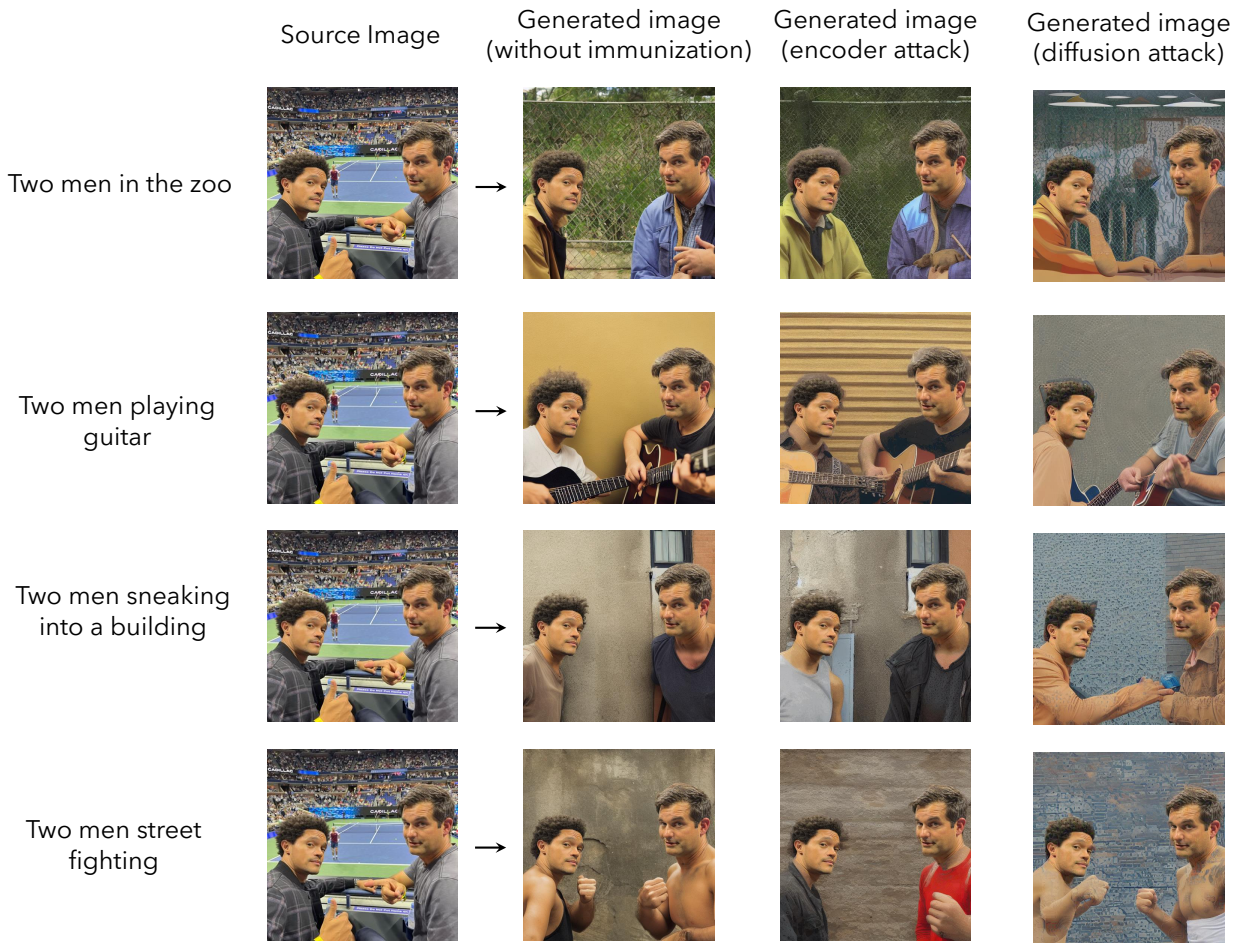

Figure 14: Immunization against image editing via prompt-guided inpainting.

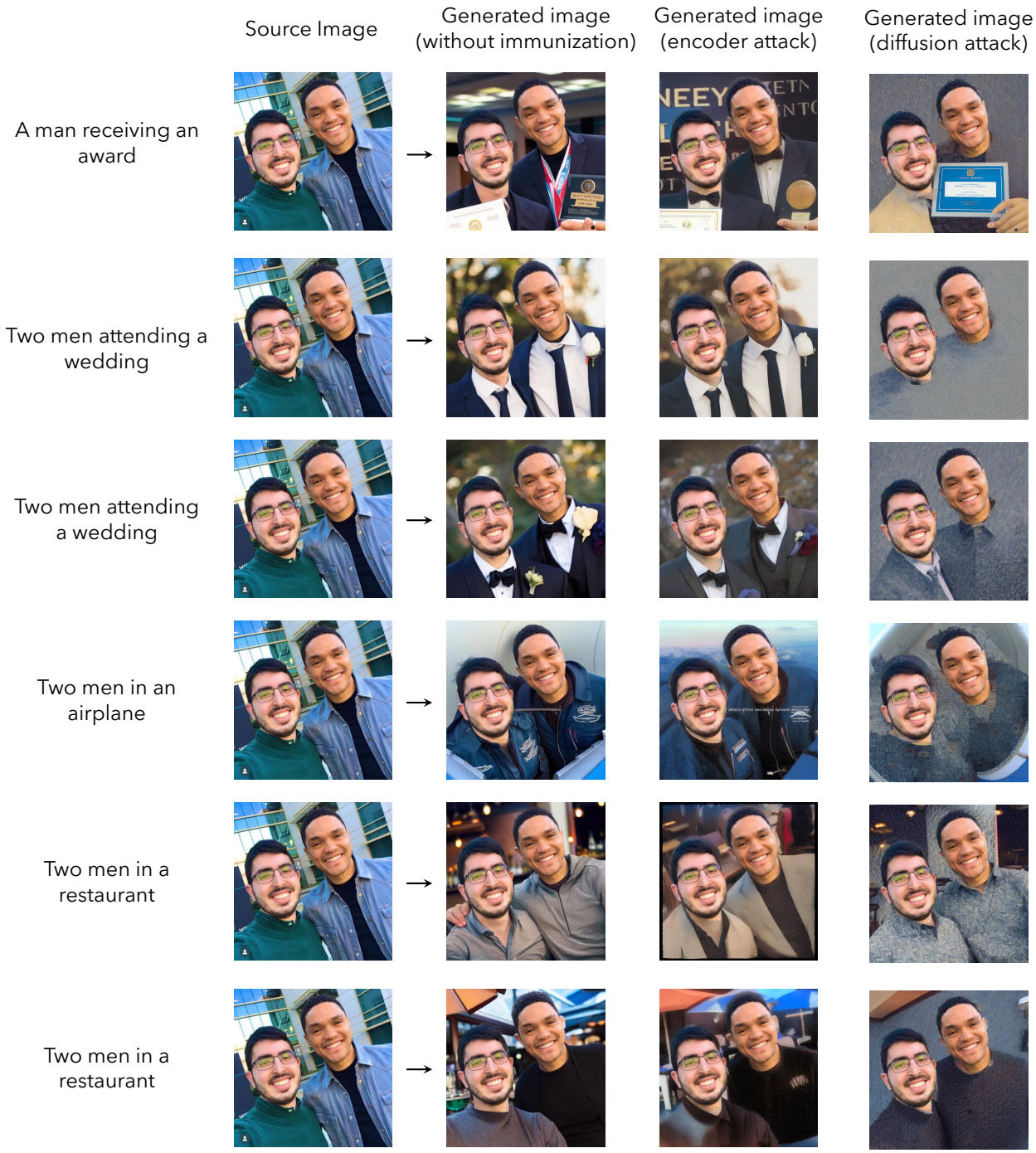

Figure 15: Immunization against image editing via prompt-guided inpainting.

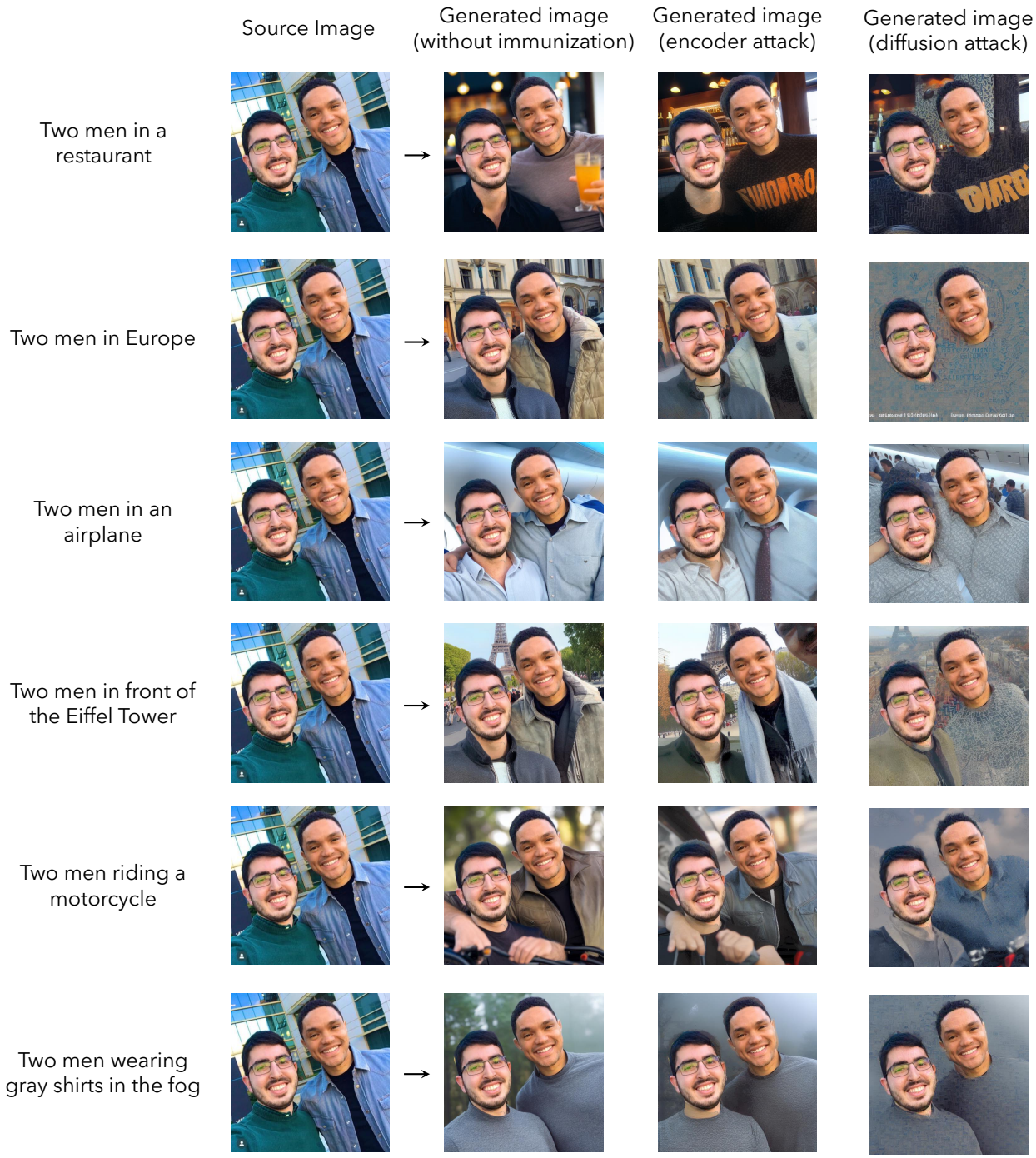

Figure 16: Immunization against image editing via prompt-guided inpainting.

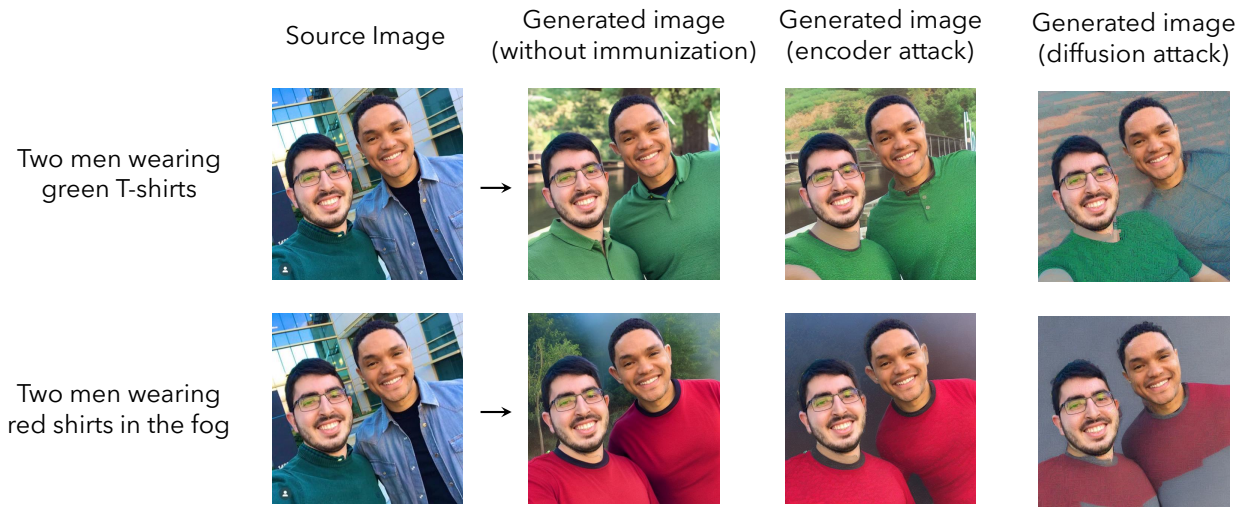

Figure 17: Immunization against image editing via prompt-guided inpainting.